Department of Computer Science & Engineering

Washington University in St. Louis
School of Engineering & Applied Science

WUCSE-2003-52

Picture Composition for a Robot Photographer

Authors: Dixon, Michael;Grimm, Cindy; Smart, William

December 1, 2002

WUCSE-2003-52: Picture Composition for a Robot Photographer

Michael Dixon Cindy M. Grimm William D. Smart

Department of Computer Science and Engineering
Washington University in St. Louis
St. Louis, MO 63130
United States

Abstract

We explain how to use simple composition rules to drive an automated, mobile photography system. The composition rules are used to determine both the location for a good photograph, and how to frame that photograph. We describe the composition component in the context of a larger application, a robotic photographer. The robot moves around an area with people in it, opportunistically looking for faces and taking photographs. We describe both how to find faces in the world and how to create "good" photographs of those faces.

Key words: Face Detection, Composition, Sensor Fusion, Mobile Robotics.

1 Introduction

Photographers employ a large set of heuristics when composing a photograph [7]. Most of these heuristics are in the form of general guidelines, and not hard-and-fast rules. We use a few of these heuristics to guide a mobile robot while it wanders around and takes pictures.

The robot, called Lewis, is a standard B21r mobile platform, from iRobot Corporation (see Figure 1). All computation and control is done on-board, using a standard Pentium-III 800MHz processor. The robot is also equipped with a pair of cameras mounted on a pan-tilt unit, a scanning laser range-finder, and contact sensors. The laser range-finder returns the distance to obstacles over the front 180° of the robot, while the contact sensors detect when the robot comes into contact with an obstacle.

We have mounted both a digital video camera and a digital still camera on the pan-tilt unit. We use the video camera to continuously scan the scene and frame the shot. When the software has determined that there is a "good" photograph, the actual photo is taken with the still camera. This two-camera system is necessary because the video camera's resolution and color are too poor for final pictures, but the still camera's transfer rate is too slow for real time processing.

The robot continuously scans the scene with both the video camera and the laser range-finder. The video frames are analyzed for faces while the laser range-finder looks for both potential obstacles and legs under the faces (see Section 3). This information is processed to produce probable locations of people, relative to the robot's current position. The system currently operates in four modes: scanning the scene, moving to a potential photograph location, framing a shot from the current location, and actually taking a picture. Section 4 discusses how we determine the best place for a photograph. Section 5 describes how we first frames a shot, and then decide if it is worth taking.

Figure 1: Lewis the robot.

We close with the results of Lewis's photography, both good and bad, in Section 6. We discuss the failures, why we think they occur, and describe some possible additions to the current system.

2 Previous work

The majority of composition-related work can be found in the virtual world, using composition rules to position a virtual camera [6, 1, 4, 11], choose a location for a virtual cinematographer [8, 2, 3, 18], or control how a scene changes [10]. In a virtual world the system has the advan-

tage of knowing where all of the individuals are, so framing a shot is a well-defined optimization problem. Also, the camera is free to move anywhere, unlike a physical camera. Like us, the on-line systems employ strategies for dealing with a constantly changing environment.

Rules from cinematography and TV interviews have also been used to control cameras in video teleconferencing [1]. There is also a growing area of research that uses a tight visual feedback loop to control robotic manipulators [9]. We share the feedback nature of these systems, but we differ in the required accuracy and on the fixed location of the camera. We do not need to position the robot or the camera with a great deal of accuracy. Our feedback loop is also operating in a relative coordinate system (the robot's position) without any knowledge of an absolute coordinate system.

Face detection and tracking has been widely studied. Approaches include simple skin detection [19, 15], learning from examples [17, 14], Eigenspace approaches [13], and template matching [12]. Somewhat close in spirit to our approach is the work by Fleck *et al* [5], which identifies patches of skin in an image, and uses heuristics about the structure of the scene to relate the skin patches to human bodies.

We are interested in an algorithm that is as fast as possible (ideally 30 frames per second), and has few false negatives (missed faces in an image). Unlike many other systems, however, we are willing to tolerate false negatives (identifying non-faces as faces), since we can post-process these out, using data from other sensors. We are also quite happy to take occasional pictures of plants, provided we find all of the people. For these reasons, we limit our face detection algorithm to looking for skin-colored blobs.

3 Face Finding

For our composition rules we need to know two things: where faces are in the current image, and the approximate location of people with respect to the robot. We assume that most of our subjects will be standing, and will be adults. Currently, we are not detecting or using any information about the direction in which people are facing.

The face detection algorithm first finds all skin-colored blobs in an image. It then relates these blobs to readings from the laser range-finder, in attempt to calculate the position and size of likely faces. Skin blobs that are the correct shape, size, and height from the ground are classified as faces. We discuss each of these steps in detail below.

3.1 Skin detection

The first step is to find skin-colored pixels in the image. It turns out that skin, even skin from different races, clusters tightly in all color spaces [19]. It has been shown that, for the type of application that we are interested in, the actual choice of color space has little impact on the performance of classification [16]. For this work, we use the UV plane of the YUV color space, which is the format in which our camera supplies images, so we do not need to do a transformation before using the data. We should reiterate at this point that we do not need perfect skin detection for our application, since we can heuristically remove false positives using data from other sensors. Using the YUV color space allows us to be fast, but without incurring many false negatives (skin patches not being identified in an image).

Figure 2 shows the portion of UV space that is classified as skin for the example image. The previous work represented the valid (skin-toned) area of UV space as an ellipse. We use a more basic representation, computing a lookup table for all possible UV values, and storing the classification (skin or not-skin) for each of these combinations. As can be seen from Figure 2, the skin-tone area is not really an ellipse, so our method allows us to define the area more precisely.

We need to train the skin detection algorithm for every new environment that the system operates in. We do this by taking a small number of images, typically five to ten, and annotating them by hand, using a simple graphical interface. For all pixels identified as skin in the GUI, the corresponding cells in the lookup table are labelled as "skin". We similarly identify all pixels that are "not-skin". Once this initial assignment is done, we blur the regions in the table, and expand them a little. This has the effect of removing noise and making the regions more contiguous. Empirically, we have found that this leads to more robust skin detection.

After the system is trained, we use the lookup table to identify areas of skin in the incoming images. As each new image comes in, we classify each pixel in it as either skin or not-skin. These classifications are then grouped together into blobs, and labelled as a potential face. At this point, we can throw out any potential faces that do not have a reasonable aspect ratio, since faces are generally taller than they are wide.

3.2 Range detection

The laser range-finder returns 180 distance readings approximately one degree apart over the front 180° of the robot. For each pixel in the camera we can find the corresponding distance reading from the laser using simple geometry. The camera pan-tilt unit is mounted in a fixed position on top of the robot. Given a pan angle we can determine which laser reading corresponds to a given pixel of the image.

The laser range-finder is not mounted directly under the camera, and this offset, O, must be accounted for by

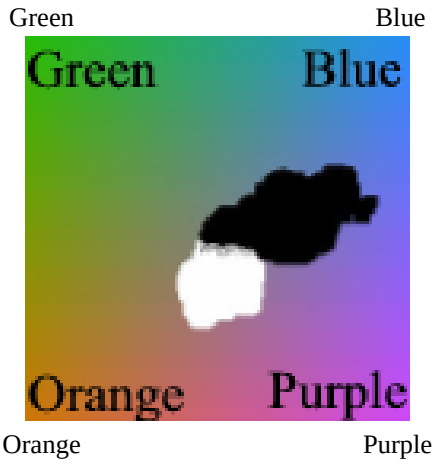
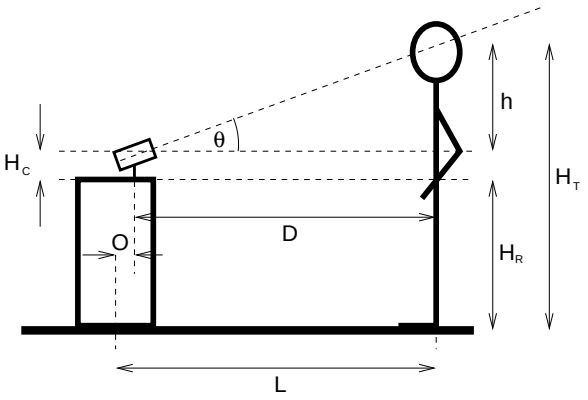


Figure 2: All possible UV colors in a sample image. The colors in the image that correspond to skin are marked as white in the UV map, all other colors as black. Colors not marked were not identified in the training image. (This figure will render as uniform grey on a black and white printer.)



$$h = D \tan\theta$$

$$D = L - O$$

$$H_T = H_R + H_C + D \tan\theta$$

Figure 3: Calculating heights and distances using the laser range-finder.

subtracting it from the reading from the range-finder, L. The height of the robot, H_R, and the height of the camera, H_C, are constants. We assume that the floor is a level plane, and that the target is standing more-or-less upright.

Once we have identified the corresponding laser range-finder readings for each candidate face, we can apply our remaining heuristics. We assume that people are standing and are between four and seven feet tall. We also assume that all faces lie within a certain size range. Based on the distance returned by the range-finder, we can calculate the height and actual size of the skin patches corresponding to the candidate faces, as shown in Figure 3. Any candidates that are outside of the height or size limits are eliminated.

4 Navigation

We use simple rules from cinematography to pick desirable photograph locations. We favor locations where the robot is approximately five feet from the subject. We do not want to take pictures where one person is occluding another. We do not want to take pictures directly down the perpendicular bisector of two people, but would rather take pictures "over" the shoulder.

To calculate the best possible position we construct an objective function, which represents the the expected quality of a picture taken from any given point. The space around the robot, which contains the people that we want to photograph, is discretized into a grid. In each grid cell, we store a number corresponding to how good a picture we think that we can take from that point. Our composition rules are written in terms of these probabilities, so it is easy to experiment with different rules and rule combinations.

4.1 Creating the objective function

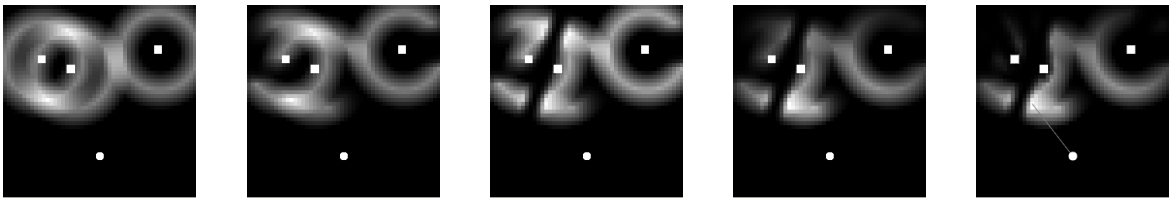
The objective function encompasses the robot's current position and the positions of people it has found. The function is initially zero everywhere, and is updated as follows.

Distance from subject (Figure 4(a)). The ideal operating distance of the still digital camera's zoom and flash is between four and seven. Therefore, the robot should be in this range for at least one of the subjects. We increase the values of the objective function in a band around each subject, with the value peaking at a distance of 5.5 feet.

Occlusion (Figure 4(b)). Locations where faces appear to overlap will not yield good photographs. For each pair of subjects, we calculate the line that runs through them, and reduce the value of the cells along that line. Cells that lie on the line segment between the two subjects, however, are left unchanged.

Bisector (Figure 4(c)). Photos taken from along a perpendicular bisector between two subjects will result in both subjects being the same distance from the camera. If they are talking to each other, this will tend to result in two profile shots, which we would like to avoid. To achieve this, we calculate the perpendicular bisector of all subjects within five feet of each other and decrease the values of the objective function along this line.

Movement (Figure 4(d)). In order to minimize the



(a) (b) (c) (d) (e)

Figure 4: Constructing the objective function to take into account (a) distance, (b) occlusion, (c) bisection, (d) movement, and (e) reachability.

chances that a photo opportunity will disappear while the robot is navigating to it, we must minimize the distance that the robot travels. On the other hand, we also wish to discourage the robot from remaining at the same location for multiple photographs. We decrease the objective function to zero for all points closer than 2 feet from the robot, or further than 20 feet. Between these extremes, we decrease the values linearly based on distance.

Reachability (Figure 4(e)). To make navigation simple, we avoid destinations which would require sophisticated path planning and obstacle avoidance. A point is considered unreachable if the robot cannot drive in a straight line to it without going through something. We use the laser range-finder information to calculate the horizon of all reachable points, and we set the objective function to zero for all points beyond this horizon.

Once the objective function is constructed, we simply look for the point with the greatest value and drive towards it. If an obstacle is encountered on the way, the destination is recalculated based on the new obstacles and the current location of faces.

5 Picture composition

In this section we describe how to pan, tilt, and zoom the camera to frame a shot based on the locations and sizes of the faces in the image. We apply different algorithms depending upon the number of people in the image (one, or more than one). The composition rules we use are standard in photography: the rule-of-thirds, the empty-space rule, the no-middle rule, and the edge rule.

Rule-of-thirds. It is best to place the faces at or near the one-third and two-thirds lines in an image (see Figure 5).

Empty space. The faces should occupy at least the middle third of the image, both horizontally and vertically.

No-middle. Do not place a single figure directly at the mid-line.

Edges. Do not place faces so they cross the boundary of the image.

The robot first pans over the scene, analyzing every frame using the techniques described in Section 3, to determine if there are any faces visible. If faces are detected, the framing algorithm must determine what subset of the faces to photograph, and how to frame the shot according to the rules of composition.

To determine the subset we first initialize it to contain the center-most face in the scene. We then find the bounding box around the faces in the subset and then calculate the ideal frame based on this bounding box using the group framing algorithm described below. We then add all faces which intersect this frame to the subset and repeat the process until no new faces are added.

If the subset found by this algorithm contains only one face, then a tighter frame, more suitable for portraits, is calculated according to the rules outlined in the single face framing algorithm. Otherwise the ideal frame is calculated with the group framing algorithm.

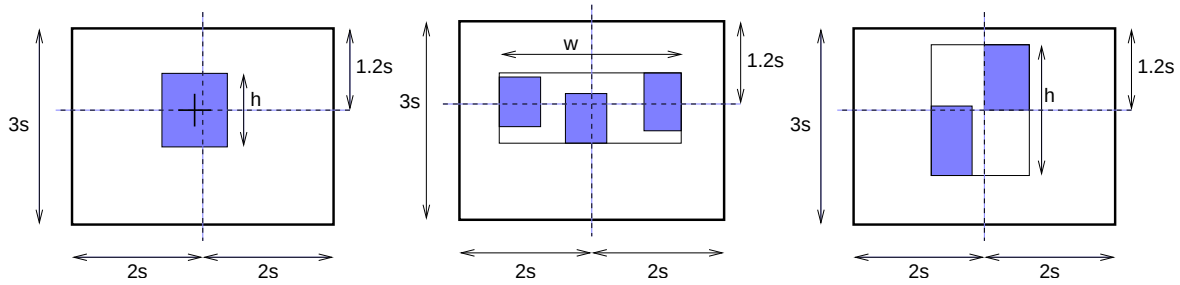
The two framing algorithms that we use are as follows.

One person. The rules applied here are the no-middle rule, the empty-space rule, and the rule-of-thirds. The ideal framing is calculated by placing the face slightly to the left or right of the center line, and ensuring that it takes up two-thirds of the image height (see Figure 6(a)). The center of the face is positioned slightly below one-third down the image. This takes into account people's hair, and necks below, which extends beyond the bounding box.

Groups. The rules applied here are the rule-of-thirds and the empty-space rule. The ideal framing is found from the width of the enclosing box for all of the faces. Again, the centerline of this box is conservatively placed slightly below one-third down the image, but is now centered in the image, as is shown in Figure 6(b). Wide groups of faces, where $w > 1.6h$, are dealt with differently than narrow ones. This

(a) (b)

Figure 5: Framing an image. (a) The large box is the optimal framing of the two detected faces (small boxes). (b) The optimal framing, showing the rule-of-thirds lines.



(a) $s = 1.5h$ (b) $s = \frac{3}{8}w$ (c) $s = \frac{1}{2}h$

Figure 6: Calculating the ideal framing for (a) a single face, and (b) a wide group of faces, and (c) a narrow group of faces.

threshold, and the ratios in Figure 6, were determined empirically to result in pleasing compositions.

The difference between the ideal frame and the current view frame represents the desired pixel-based changes. We convert from pixels to radians according to our camera model and set our pan, tilt, and zoom parameters according to these values. This framing process runs continuously. As the camera is adjusting, the incoming images are constantly analyzed and the target frame repositioned.

The final item to discuss is when to actually take a picture. If all subjects were completely stationary then we could take the time to make the desired frame exactly match the picture boundary. Unfortunately, people move. We also want to take pictures as often as possible, even if the results are not perfect. We therefore use a decaying threshold. The longer the robot has spent framing a shot, the more slop is allowed in the framing box, until a photograph is eventually taken.

6 Results and conclusions

In this section we show some of the best and some of the worst photographs the robot has taken (Figures 7 and 8). The majority of the photographs taken contain faces; however, the robot does have a fondness for yellow posters and pink flowers. As an anecdotal measure of success, 1000 of the 3000 photographs Lewis took at SIGGRAPH 2002, were selected by visitors for emailing or printing.

There are, however, a number of common failure modes for the system. All of these are the subject of current work.

People moving. It takes approximately five seconds for Lewis to zoom the video camera and frame a picture. People sometimes move away or turn away before Lewis has finished framing the shot, causing the robot to either not take a picture or have someone walk across or out of the frame.

Inability to frame a shot. Sometimes Lewis spends several seconds attempting to frame a shot and never succeeding, even when the people are not moving. This can happen for several reasons. For

example, the face detection algorithm may create different blobs depending on how the camera is zoomed, or the person may be wearing a red shirt which gets classified as skin.

This problem can also be caused by faces whose pixels fall close the the skin/not-skin boundary in UV space. Due to small variations, these faces may be detected in one frame, but not the next, causing the current algorithm to loop.

Wrong assumptions. In order to simplify face-finding, we make some assumptions about the world which are not always true. For example, the shortest range-finder reading under a potential face does not always correspond to a person's legs. It might be a completely unrelated object on the floor, and the potential face might be a large poster, off in the distance. If the geometry conspires against us, this will be classified as a face, and become part of a composition.

In general, we can avoid such mistakes by calculating things more directly from the world, rather than inferring them. In the above example, we plan to add a second camera, and detect the approximate depth of face candidates using stereo vision techniques. This makes fewer assumptions about the structure of the world, and will solve some of the current problems.

In this paper, we have described how simple composition rules can be encoded, and used to drive a mobile robot platform as it takes photographs. All of the rules we use are simple, but in combination, they seem to be able to detect, compose, and take reasonably good pictures. The system is certainly nowhere near a human photographer, but it does provide an interesting platform for research into automatic composition, and robot control.

Acknowledgements

This work was supported in part by NSF REU award #0139576, and NSF award #0196213.

References

[1] W. H. Bares and J. C. Lester. Cinematographic user models for automated realtime camera control in dynamic 3D environments. In *Proceedings of the Sixth International Conference on User Modeling*, pages 215–230, 1997.

[2] Steven M. Drucker, Tinsley A. Galyean, and David Zeltzer. Cinema: A system for procedural camera movements. In *1992 Symposium on Interactive 3D Graphics*, volume 25, pages 67–70, March 1992. ISBN 0-89791-467-8.

[3] Steven M. Drucker and David Zeltzer. Camdroid: A system for implementing intelligent camera control. In *1995 Symposium on Interactive 3D Graphics*, pages 139–144. ACM SIGGRAPH, April 1995. ISBN 0-89791-736-7.

[4] Shachar Fleishman, Daniel Cohen-Or, and Dani Lischinski. Automatic camera placement for image-based modeling. *Computer Graphics Forum*, 19(2):101–110, 2000.

[5] David Forsyth and Margaret Fleck. Automatic detection of human nudes. *International Journal of Computer Vision*, 32(1):63–77, 1999.

[6] Bruce Gooch, Eric Reinhard, Chris Moulding, and Peter Shirley. Artistic composition for image creation. *Eurographics Workshop on Rendering*, 2001.

[7] Tom Grill and Mark Scanlon. *Photographics Composition*. American Photographics Book Publishing, 1990.

[8] Li-Wei He, Michael F. Cohen, and David H. Salesin. The virtual cinematographer: A paradigm for automatic real-time camera control and directing. *Computer Graphics*, 30(Annual Conference Series):217–224, 1996.

[9] S. Hutchinson, G. Hager, and P. Corke. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automattion*, 12(5):651–670, 1996.

[10] Michael Kowalski, John Hughes, Cynthia Rubin, and Jun Ohya. User-guided composition effects for art-based rendering. *ACM Symposium on Interactive 3D Graphics*, 2001.

[11] Éric Marchand and Nicolas Courty. Image-based virtual camera motion strategies. In *Graphics Interface*, pages 69–76, 2000. ISBN 1-55860-632-7.

[12] N. Oliver, A. Pentland, and F. Berard. Lafter: Lips and face real time tracker. In *Proc. Computer Vision and Patt. Recog.*, 1997.

[13] A. Pentland, B. Moghaddam, and T. Starner. View-based and modular eigenspaces for face recognition. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR'94)*, Seattle, WA, June 1994.

[14] Henry A. Rowley, Shumeet Baluja, and Takeo Kanade. Human face detection in visual scenes. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 875–881. The MIT Press, 1996.

[15] K. Schwerdt and J. Crowley. Robust face tracking using color. In *Proc. of 4th International Confer-*

Figure 7: Good pictures.

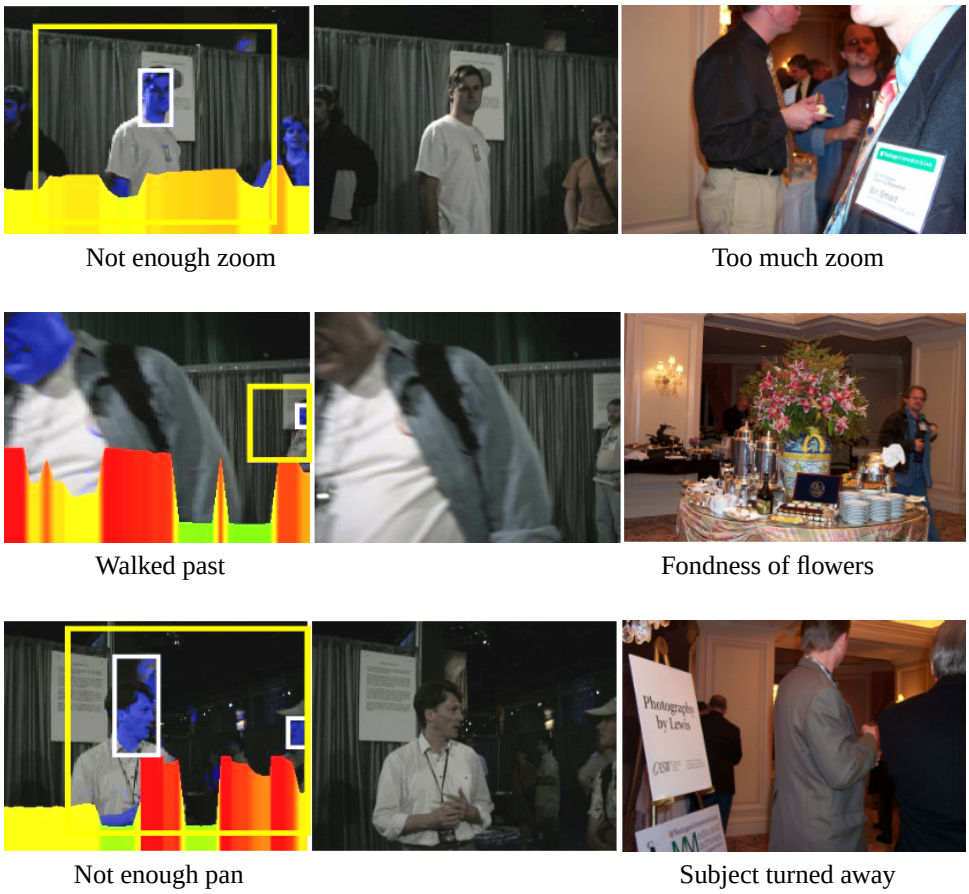


Figure 8: Bad pictures.

ence on Automatic Face and Gesture Recognition, pages 90–95, 2000.

[16] Min C. Shin, Kyong I. Chang, and Leonid V. Tsap. Does colorspace transformation make any difference on skin detection? In *Proceedings of the IEEE Workshop on Applications of Computer Vision*, Orlando, FL, December 2002.

[17] Kah Kay Sung and Tomaso Poggio. Example-based learning for view-based human face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):39–51, 1998.

[18] Bill Tomlinson, Bruce Blumberg, and Delphine Nain. Expressive autonomous cinematography for interactive virtual environments. In Carles Sierra, Maria Gini, and Jeffrey S. Rosenschein, editors, *Proceedings of the Fourth International Conference on Autonomous Agents*, pages 317–324, Barcelona, Catalonia, Spain, 2000. ACM Press.

[19] Jie Yang, Weier Lu, and Alex Waibel. Skin-color modeling and adaptation. In *ACCV (2)*, pages 687–694, 1998.